REMARKS

Pending claims

Assuming entry of this amendment, claims 1-45 are pending, of which claims 1, 4, 13, 25, 26, 27, 28, 30, 34, 39, 44 and 45 are independent.

Specification Amendment

The change (other than the correction of a typographical error) requested in the specification relates to the concept "virtual machine monitor." As stated in the last sentence of paragraph [0006]: "The implementation and general features of a VMM are known in the art." Thus, this paragraph is summarizing certain known features of virtual machine monitors. One known feature of modern VMMs is that it is not necessary (although frequently implemented) for a VMM to virtualize all the hardware resources of the hardware platform on which it runs; rather, it may virtualize only a subset, or, indeed, a superset, of the resources, and these need not relate to the actual hardware present. The change to the specification is requested in order to reflect this known property; it adds no new matter, but rather simply better describes known VMMs.

Claim Rejections – Rejections Under 35 U.S.C. 103(a)

The Examiner rejected claims 1-3,24,28 and 29 under 35 U.S.C. 103(a) as being unpatentable over Bonola (U.S. Patent 5,913,058 – "*Bonola*") in view of Solomon (U.S. Patent 6,269,409 B1 – "*Solomon*"). Of these, claims 1 and 28 are independent.

In particular, tracking claim 1, the Examiner wrote that *Bonola* teaches all the features of claim 1 (and 28) except the feature of the kernel substantially displacing the COS.

As the specification makes clear, the applicants' invention may be used not only in multi-processor systems, but also in those with only a single CPU. (See, for example, [0106]: "The kernel 600 according to the invention may also easily be configured to support more than one CPU 110, although it of course may also run on single-CPU systems."

In contrast, *Bonola* presupposes at least two processors, since *Bonola* involves arranging one of a plurality of processors to operate as a dedicated I/O processor 12d. The real-time kernel 30 then executes on this dedicated I/O processor (col. 5, lines 9-12). In *Bonola*, once the kernel 30 is loaded, the host operating system (host OS) remains in control of access to all other system resources, in particular, to the remaining processors 12a-12c. See, for example, col. 10, lines 3-8:

> After spawning the real-time kernel 30 on the I/O processor 12d and being notified by the real-time kernel 30 that the real-time kernel 30 has completed its initialization, the loader program 32 returns control back to the host operating system 26.

As *Bonola* then explains in detail throughout the rest of the subsequent text, the host OS continues to function as a conventional operating system in that it continues to control access to all system resources except the I/O processor 12d and its associated device 24. The host OS will execute on the remaining processors 12a-12c as in any other conventional multi-processor computer, with no other software layer scheduling its access to these processors.

Bonola thus discloses a system in which an I/O device 24 is "assigned" a dedicated processor, with accompanying system software (the kernel 30), so as to reduce the "large processing bandwidth burden on the system processors" (col. 1, lines 34-35). *Bonola* does this in a way (loading the kernel before other device drivers are loaded during booting of the host operating system) intended to avoid the conflicts that would arise if the host operating system (including drivers installed during boot-up) were to attempt to access the dedicated processor directly too soon (col. 2, lines 30-41). In essence, *Bonola* converts the I/O device 24 into a much higher speed device by "giving" it one of the processors 12a-12d that would otherwise be available to the host operating system. When both the host operating system and the kernel 30 are operational, the host operating system continues to execute as it would in any other system, directly accessing any physical resources of the computer except the I/O device 24 and the dedicated processor 12d.

The Examiner correctly noted (point 4 of the Office action): "*Bonola* does not explicit[ly] teach the kernel substantially displacing the COS." To clarify the concept of

OS displacement according to the invention, claims 1 and 28 have been amended to recite that displacement by the kernel of the COS involves, in general, that the kernel then handles requests for system resources (as in the original claims), and, in particular, that the kernel then also handles "<u>scheduling execution of the COS on the hardware processor(s)</u>."

On the other hand, the Examiner went on to write that *Solomon* teaches the kernel substantially displacing the COS and that it would be obvious to combine the teachings of *Bonola* and *Solomon* "because Solomon's Window NT operating system to the base machine (sic) into a format that is processed by the Unix operating system would allow for execution for multiple operating systems depend on the emulation of one environment or the other and provide an improvement for concurrently executing multiple operating systems."

In support of this, the Examiner pointed to *Solomon* col. 7, lines 28-37, which state:

> Interaction between the Windows NT operating system and the base machine are handled by the UNIX operating system through a translation of requests...and calls from a format normally made by the Windows NT operating system to the base machine into a format that is processed by the UNIX operating system. Request and data intended for the Windows NT operating system are received by the UNIX operating system and translated by the software abstraction layer into a format usable by the Windows NT operating system.

Not only does this citation not support the assertion, but rather it demonstrates why *Solomon* also fails to disclose displacement of a native OS by another software entity (such as the applicants' kernel) such that the other software entity takes over the task of scheduling execution of the native OS on the underlying hardware processor(s) in the base machine: In *Solomon*, the first operating system (Fig. 3: "Native Unix" 306; Figs. 5, 6 and 7: "Unix OS" 506, 606, and 706, respectively) *always* is the operating system that handles requests for the resources of the base machine 302, 502, 602, 702, including requests from the second operating system (Fig. 3: "NT" 314; Figs. 5, 6 and 7: "Windows NT OS" 508, 614, and 708, respectively), which is *never* able to directly access the base machine or schedule its own execution. Indeed, the whole point of *Solomon* is to provide a mechanism so that the second operating system can

access the base machine at all in an environment where the commands of the first, native (Unix) operating system differ from those of the second. Thus:

> Col. 2, lines 20-25 – also cited by the Examiner:
> Interaction between the second operating system and the base machine is handled by the software abstraction layer translating requests and calls from a format normally made by the second operating system to the base machine into a format that is processed by the first operating system. Request and data intended for the second operating system are received by the first operating system, sent to the software abstraction layer and translated by the software abstraction layer into a format usable by the first operating system.
>
> Col. 3, lines 54-57:
> Operating system 314 does not have a direct interface to base machine 302. Instead, software abstraction layer (SAL) 320 is employed as interface between operating system 314 and operating system 316.
>
> Col. 4, lines 29-32:
> In accordance with the preferred embodiment of the present invention, the hardware abstraction layer, interface 402 is replaced with a software abstraction layer to allow Windows NT operating system 400 to execute within system 300 in FIG. 3.

In contrast, as explained in amended independent claims 1 and 28, the first operating system (COS) initially is "installed to run on the hardware processor at a most-privileged, system level, the system level being defined as an operational state with permission to directly access predetermined physical resources of the computer," but once the kernel is loaded, the kernel also takes responsibility for the task of "<u>scheduling execution of the COS on the hardware processor(s)</u>."

This is explained at length in the specification, but is mentioned, for example, in paragraphs [0016] and [0044] of the specification:

> [0016] In the preferred embodiment of the invention ... [t]he kernel thereby separately schedules the execution of the COS and of each VM.
>
> [0044] In conventional computer architectures, the operating system is at system level. As shown in Figure 1, the OS 420 is not; rather, a software module referred to here as the "kernel" 600 is interposed between the OS 420 and the hardware platform. Thus, the kernel 600 may be viewed either as displacing the OS 420 from the system

level and taking this place itself, or as residing at a "sub-system level." When interposed between the OS 420 and the hardware 100, the kernel 600 essentially turns the OS 420 into an "application," which has access to system resources only when allowed by the kernel 600, which schedules the OS 420 as if it were any other component that needs to use system resources. For this reason, the OS 420 is referred to here in places as the "console" or the "console OS" or simply the "COS" 420.

Solomon's second OS (Windows NT) is *isolated* from the base machine by the first (Unix) OS, but it is never *displaced* by the first OS, since it *never* schedules its own execution on the hardware processor – its commands must at all times be converted from its own format into the Unix OS format. The first, native Unix OS must therefore always be resident and executing for the second, Windows OS to function at all.

Neither *Bonola* not *Solomon* therefore discloses OS displacement as defined in the independent claims 1 and 28. Of course, no combination of these two references would provide a method and system that have features neither reference discloses. Moreover, it would not be obvious to combine *Bonola* and *Solomon* in any event, since the whole idea of *Bonola* is to allow a form of kernel 30 to have direct access to the I/O processor 12d of the "base machine" whereas the whole idea of *Solomon* is to provide a way for a secondary OS to communicate with the base machine *indirectly*, via a special software abstraction layer (see, for example, *Solomon* col. 3, lines 54-57 cited above).

Independent claims 1 and 28 thus define the invention to have a feature of OS displacement not found in either of the cited references. These claims should therefore be allowable over this cited prior art, as should their respective dependant claims, which of course inherit their limitations. This feature is likewise not found in any of the other cited prior art used to reject various ones of the dependent claims.

Allowable Subject Matter

Claims objected to but allowable if rewritten

The Examiner objected to Claims 4-11, 13-20, 25, 30-32, 34-38, 39-40 as being dependent upon rejected base claims, but indicated that they would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims. Claims 4, 13, 25, 30, 34 and 39 have be rewritten in independent form so as to include limitations as follows:

Independent Claim		Base claim	Intervening claim(s)
4	now also includes the limitations of:	1	2
13	"	1	12
25	"	1	24
30	"	28	--
34	"	28	33
39	"	28	33

These independent claims should now be allowable, as should the claims that depend from them.

Allowed claims

The Examiner allowed claims 26, 27, 44, and 45. Claims 27 and 45 have been amended, however, to eliminate a typographical error: The plural of VMM should be --VMMs-- instead of --VMM's--.

Other claim amendments

The applicants have amended some of the claims not because of the cited prior art, but rather for the sake of clarity and consistency both internally and with the specification and to eliminate a typographical error.

Original claims 2-25 are method claims, but their base claim 1 was written as a system claim. This was a cut-and-paste error. Claim 1 has consequently been

amended to ensure that it properly recites a method that is practiced in the context of a computer system rather than as a computer system itself. This correction has been incorporated into newly independent claims 4, 13, and 25 as well.

In claims 28-31, 34, 35, 37, 39, 41, and 43, the terms "kernel means" and "loading means" have been replaced by simply "kernel: and "loader," since these are the terms actually used in the specification .

Conclusion

The independent claims recite advantageous features of the invention that are not found at all in the newly cited reference. As such, the independent claims should be allowable over the cited prior art. The various dependent claims of course simply add additional limitations and should therefore be allowable along with their respective independent base claims.

Change of Attorney Name

Please note the enclosed letter concerning the change of the attorney's name from "Slusher" to "Pearce."

Change of Payment Status – No Longer Small Entity

Please note the enclosed letter giving notice that the applicant is no longer a small entity for purposes of paying reduced fees.

Date: 20 September 2004

Respectfully submitted,

Jeffrey Pearce

34825 Sultan-Startup Rd.
Sultan, WA 98294
Phone & fax: (360) 793-6687

Jeffrey Pearce
Reg. No. 34,729
Attorney for the Applicants